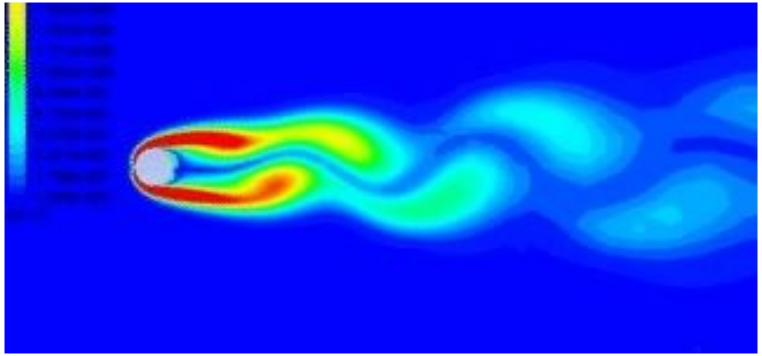
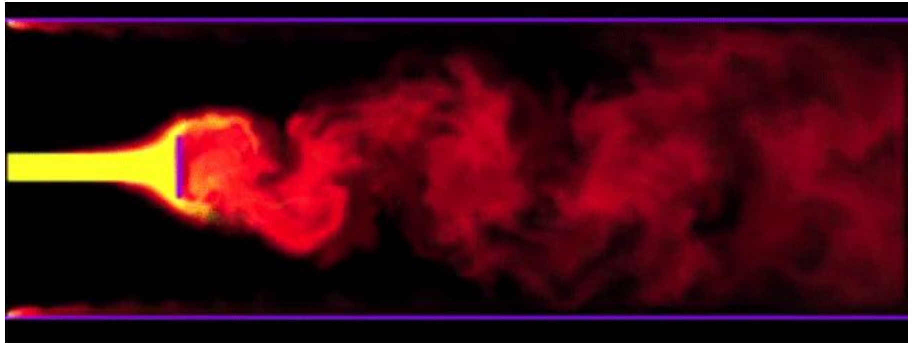
PhiFlow Python Package for Fluid Simulations

Toolbox 1: Antoine Assaf

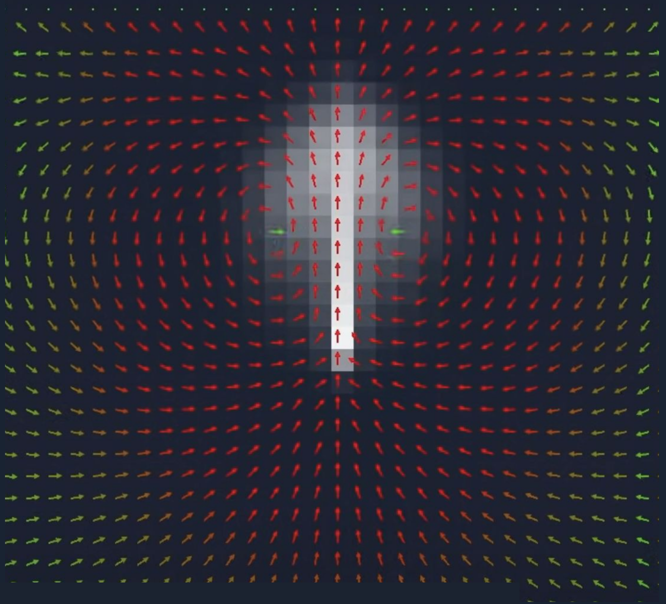
What is PhiFlow?

- A physics-based simulation library for fluid dynamics and other PDE-based problems
- Python package built upon PyTorch, Jax, and TensorFlow
- Supports Eulerian (grid-based) and Lagrangian (particle-based) simulations...
- Will focus on the Grid-based simulation

BTS: Navier-Stokes Equation

1. Define a Domain (grid resolution, size, boundaries)
2. Initialize fields (velocity, pressure, smoke, etc.).
3. Apply forces (gravity, inflow, obstacles).
4. Solve the Navier-Stokes equations (advection, diffusion, pressure projection).
5. Visualize or export the results.



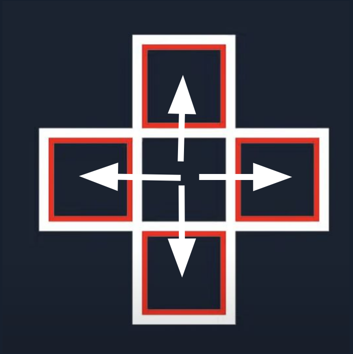
$$\nabla \cdot \mathbf{u} = 0 \qquad \text{(incompressibility)}$$

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla p + \mu \nabla^2 \mathbf{u} + \rho \mathbf{F}$$

Momentum = -pressure gradient + viscosity + external forces (gravity, walls, wind)

Background Info: Grid Components & Properties

Diffusion



Distribution of density per time step

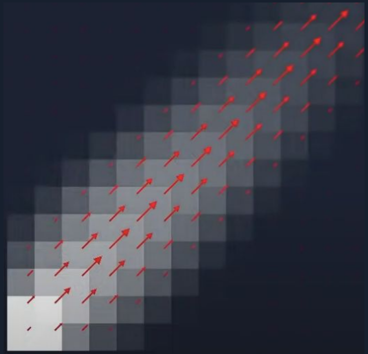
PhiFlow built-in support for diffusion through its iterative physics solvers

diffuse(diffusion_rate): Applies diffusion with the specified rate.

$$d_n = \frac{d_c + k s_n}{1 + k}$$

dn = next density step
dc = current density
k = rate-related constant

Advection



Moving smoke along velocity vectors through domain

Semi-Lagrangian: Stable, allows large time steps, but causes blurring.

MacCormack: More accurate, less diffusion, but can overshoot.

$$\frac{\partial s}{\partial t} + (u \cdot \nabla) \cdot s = \alpha \nabla^2 s + i$$

s = concentration of smoke
u = velocity vector
alpha = diffusivity
i = inflow

Personal Attempt tackling PhiFlow

Volleyball “Float Serve” A serve with little-to-no spin which snakes through the air to confuse receivers

Ran a simulation in PhiFlow to visualize the turbulent behavior from this phenomenon that causes the ball to knuckle

Approach: Initialized a standard velocity and smoke field (smoke field can be thought as density field)

Created an SoftGeometryMask obstacle mask representing the volleyball with 0 velocity

Created Inflow “spawns” with rightward acceleration to model air flowing against the volleyball, hopefully with turbulent behavior behind the ball.

Float Serve Simulation Set Up

Adding velocity and smoke grids

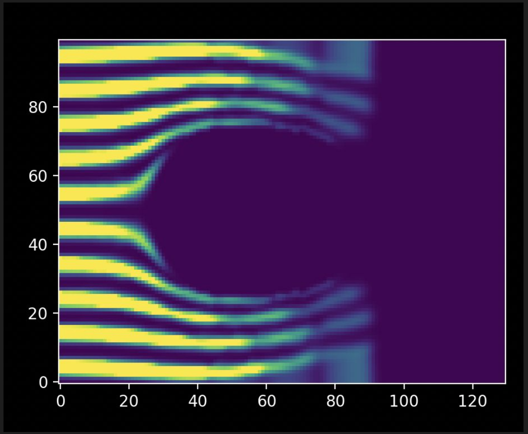
```
# Initialize velocity field (start with zero velocity)
velocity = flow.StaggeredGrid(
    values=(5.0, 0),
    extrapolation=flow.extrapolation.BOUNDARY,
    x=130,
    y=100,
    bounds=flow.Box(x=130, y=100),
)

# Initialize smoke starting with 0 everywhere
smoke = flow.CenteredGrid(
    values = 0.0,
    extrapolation=flow.extrapolation.BOUNDARY,
    x=130,
    y=100,
    bounds=flow.Box(x=130, y=100),
)
```

Creating volleyball mask (a circular wall that the smoke collides with)

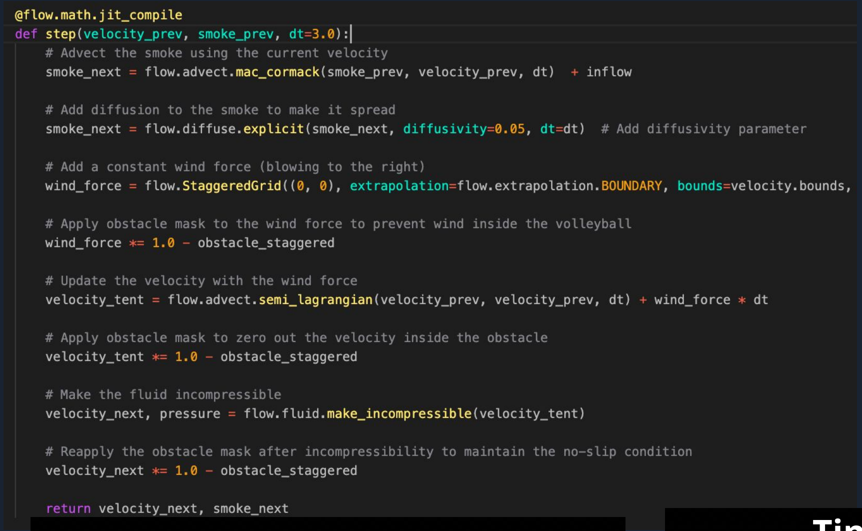
```
# Define the volleyball obstacle (a circle in the middle of the screen)
obstacle_mask = flow.SoftGeometryMask(
    flow.Sphere(x=50, y=50, radius=25)  # Center at (50, 50) with radius 10
)

# Convert the obstacle mask into a CenteredGrid (same resolution as velocity)
obstacle_grid = flow.CenteredGrid(
    values=obstacle_mask,
    extrapolation=flow.extrapolation.ZERO,  # Use zero extrapolation for the obstacle
    bounds=velocity.bounds,
    resolution=velocity.resolution,
)

# Convert obstacle_grid to match the staggered velocity grid
obstacle_staggered = obstacle_grid.at(velocity)
```

Inflow spawns (simulating wind)

```
inflow = 1 * flow.CenteredGrid(
    values=flow.SoftGeometryMask(
        flow.union(
            [
                flow.Sphere(x=0, y=5, radius=1),
                flow.Sphere(x=0, y=15, radius=1),
                flow.Sphere(x=0, y=25, radius=1),
                flow.Sphere(x=0, y=35, radius=1),
                flow.Sphere(x=0, y=45, radius=1),
                flow.Sphere(x=0, y=55, radius=1),
                flow.Sphere(x=0, y=65, radius=1),
                flow.Sphere(x=0, y=75, radius=1),
                flow.Sphere(x=0, y=85, radius=1),
                flow.Sphere(x=0, y=95, radius=1),
            ]
        )
    ),
    extrapolation=0.0,
    bounds=smoke.bounds,
    resolution=smoke.resolution,
)
```

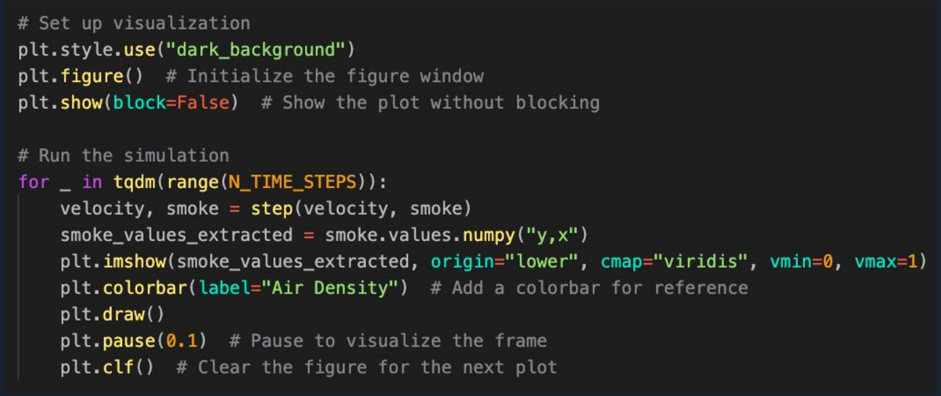


Applying Diffusion and Advection formulas per step

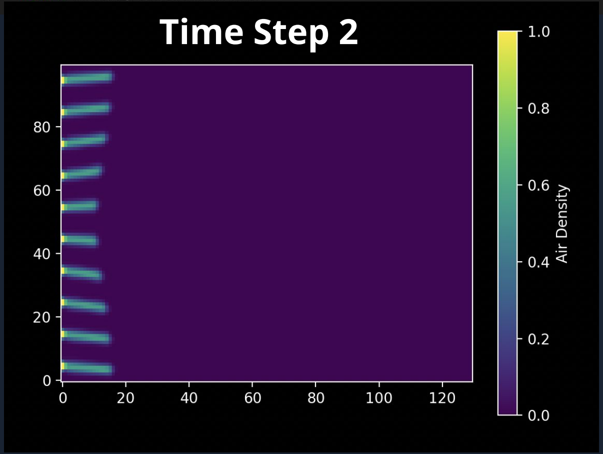


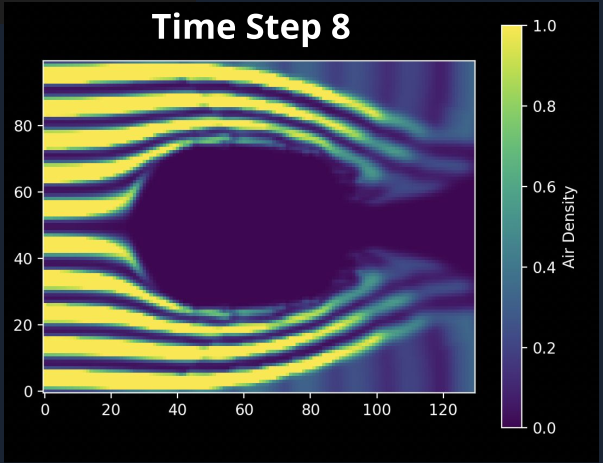
```python
@flow.math.jit_compile
def step(velocity_prev, smoke_prev, dt=3.0):
    # Advect the smoke using the current velocity
    smoke_next = flow.advect.mac_cormack(smoke_prev, velocity_prev, dt)  + inflow

    # Add diffusion to the smoke to make it spread
    smoke_next = flow.diffuse.explicit(smoke_next, diffusivity=0.05, dt=dt)  # Add diffusivity parameter

    # Add a constant wind force (blowing to the right)
    wind_force = flow.StaggeredGrid((0, 0), extrapolation=flow.extrapolation.BOUNDARY, bounds=velocity.bounds,

    # Apply obstacle mask to the wind force to prevent wind inside the volleyball
    wind_force *= 1.0 - obstacle_staggered

    # Update the velocity with the wind force
    velocity_tent = flow.advect.semi_lagrangian(velocity_prev, velocity_prev, dt) + wind_force * dt

    # Apply obstacle mask to zero out the velocity inside the obstacle
    velocity_tent *= 1.0 - obstacle_staggered

    # Make the fluid incompressible
    velocity_next, pressure = flow.fluid.make_incompressible(velocity_tent)

    # Reapply the obstacle mask after incompressibility to maintain the no-slip condition
    velocity_next *= 1.0 - obstacle_staggered

    return velocity_next, smoke_next
```

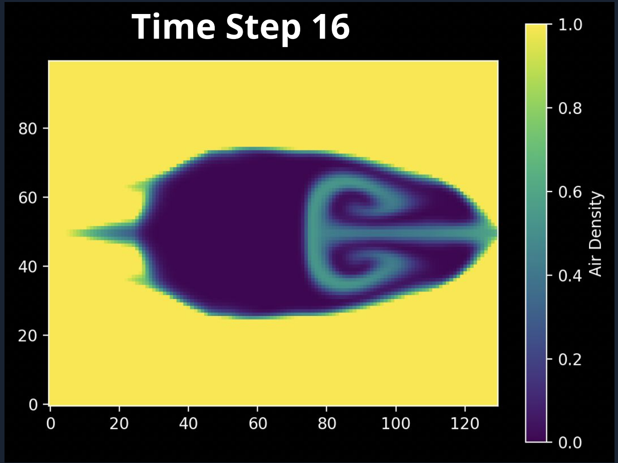
Displaying animation with matplotlib



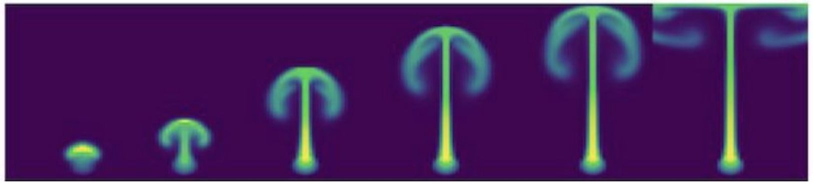
```python
# Set up visualization
plt.style.use("dark_background")
plt.figure()  # Initialize the figure window
plt.show(block=False)  # Show the plot without blocking

# Run the simulation
for _ in tqdm(range(N_TIME_STEPS)):
    velocity, smoke = step(velocity, smoke)
    smoke_values_extracted = smoke.values.numpy("y,x")
    plt.imshow(smoke_values_extracted, origin="lower", cmap="viridis", vmin=0, vmax=1)
    plt.colorbar(label="Air Density")  # Add a colorbar for reference
    plt.draw()
    plt.pause(0.1)  # Pause to visualize the frame
    plt.clf()  # Clear the figure for the next plot
```





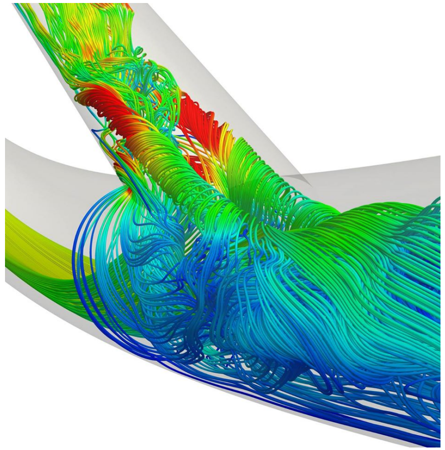


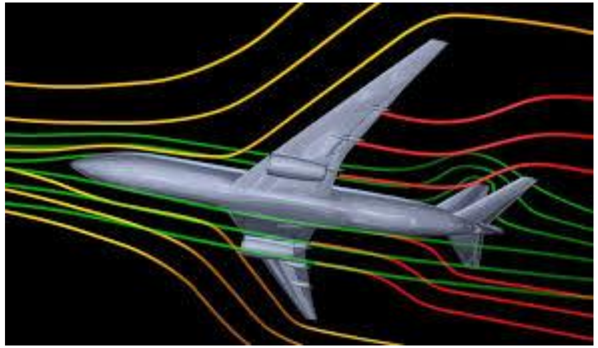
Current Applications in Physics



An application involving Differentiable Physics? PhiFlow is useful

- **Aerodynamics** Simulating airflow over wings, cars, and other objects
- **Ocean and Tsunami Simulations** – Studying wave propagation and coastal impact
- **Blood Flow Simulation** – Modeling vascular fluid dynamics for medical diagnostics
- **Buoyancy-driven flow** - like a smoke plume





Sources

YT: Machine Learning & Simulation

https://www.youtube.com/watch?v=KMfcF9XvVio&list=PLYLhRkuWBmZ5R6hYzusA2JBIUPFEE755O&index=5

This YT series from the creator of PhiFlow himself:
https://www.youtube.com/watch?v=YRi_c0v3HKs&list=PLYLhRkuWBmZ5R6hYzusA2JBIUPFEE755O&index=2

But how DO Fluid Simulations Work?
https://www.youtube.com/watch?v=qsYE1wMEMPA

DeepSeek LLM for helping with code